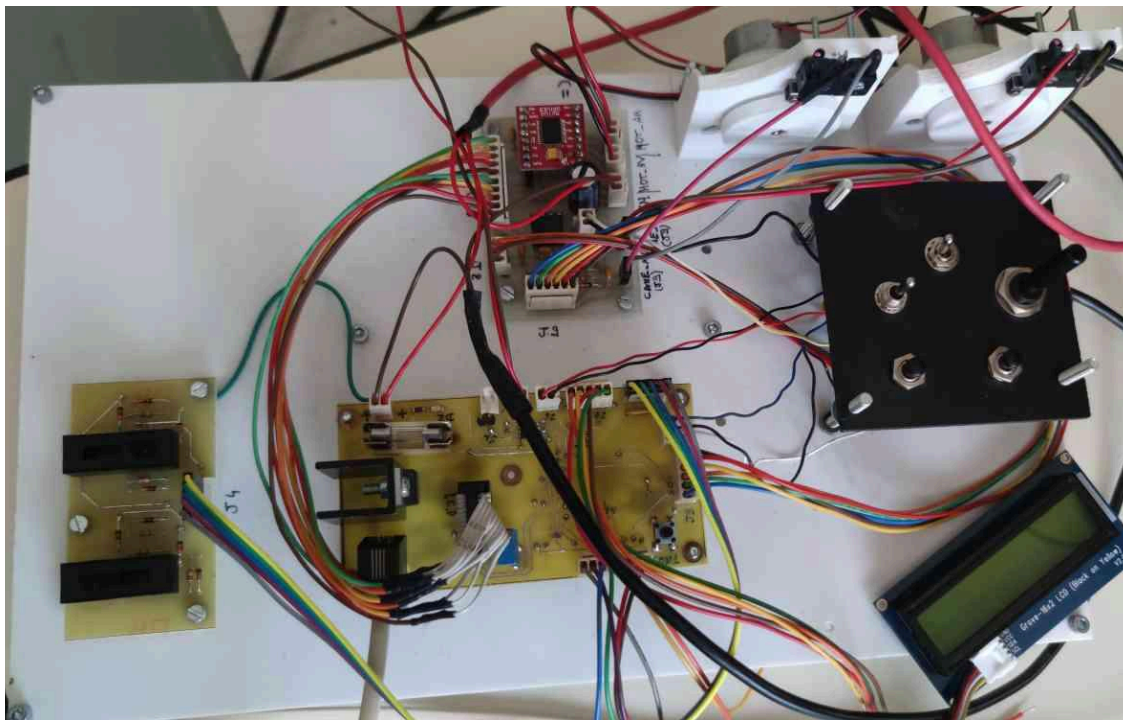




### Plan :

- Présentation du projet.
- Cahier des charges.
- Étude de la carte d'un point de vue électronique afin de déterminer les entrées et les sorties.
- Programmation du projet .



# 1. Présentation du projet et de son cahier des charges.

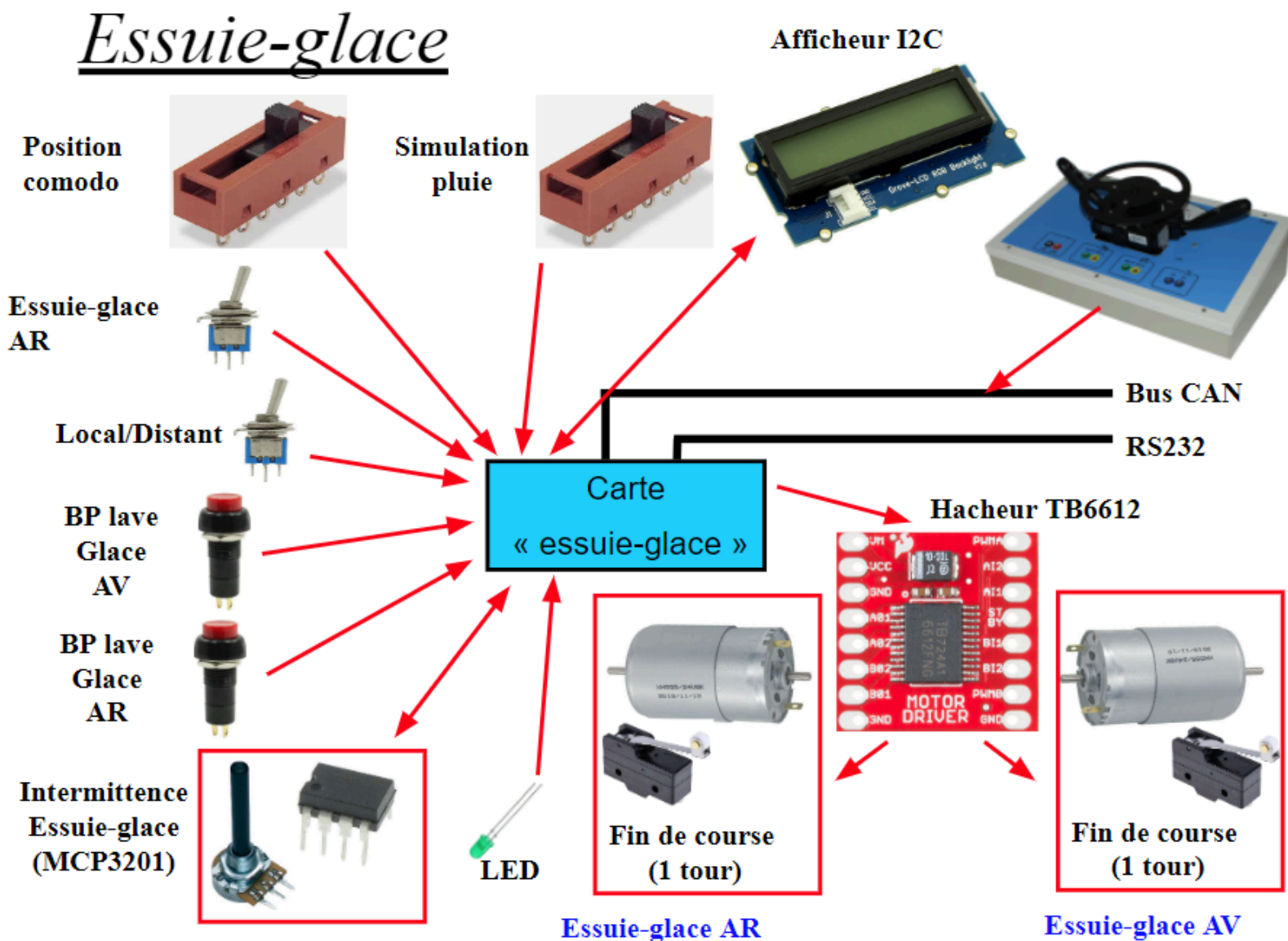
Le projet est de réaliser un contrôle simplifié d'une voiture a partir de six cartes (Feux avant, Feux arrière, Essuie-glace, Ordinateur de Bord, BSI et Calculateur) qui seront partagé par 5 groupes composés de deux étudiants ou trois étudiants au maximum.

Nous avons choisie la carte Essuie-glace qui a pour fonction de réaliser un fonctionnement d'un essuie glace en rajoutant deux modes un mode locale qui fonctionne avec des boutons et des switches afin de simuler une demande d'un utilisateur par exemple activer l'essuie glace arrière et avant en position rapide .Le mode local nous est particulièrement utile pour réaliser les fonctions de bases d'un essuie glace sans avoir les contraintes d'utiliser pour le bus can et de contrôler ses trames .

Le deuxième mode est le mode distant qui sera utilisé avec le bus can et le volants pour faire de réel test sur le projet .

Le capteur de pluie sera le même pour les deux modes il sera représenté par un switch à 4 positions .

En résumée :



- Récupère les commandes du commodo (COM2000)
- Mode de contrôle local (par un interrupteur) :
  - - BP lave glace (Push/fixe),
  - - position comodo essui glace,
  - - simulation capteur de pluie.
- Contrôle via un hacheur (TB6612 Sparkfun) deux motoréducteurs CC pour les essuie-glaces AV et AR dotés de 2 capteurs de détection d'un tour.
- Réglage intermittence par potentiomètre (ADC MCP3201)
- Communique sur un Bus CAN.
- Informe par une RS232 pour la mise au point.

## 2. Cahier des charges.

Réaliser un mode local/distance à l'aide d'un interrupteur à levier.

- **Mode local :**

Pour simuler la position des comodos nous utiliserons des interrupteurs glissière à 4 positions.

**POSITION 0:** Aucun mode sélectionnée donc faire le traitement que des laves glace AV/AR et éviter que les essuie glace AV et AR obstrue la vue de l'utilisateur.

**POSITION 1:**Sélection du mode AUTOMATIQUE donc prendre en compte le capteur de pluie

MODE SÉLECTIONNE AUTOMATIQUE PAS DE PLUIE si le capteur de pluie est en position 0

MODE SÉLECTIONNE AUTOMATIQUE PLUIE FAIBLE si le capteur de pluie est en position 1

MODE SÉLECTIONNE AUTOMATIQUE PLUIE MOYENNE avec potentiomètre si le capteur de pluie est en position 2

MODE SÉLECTIONNÉ AUTOMATIQUE PLUIE FORTE si le capteur de pluie est en position 3

**POSITION 2:** Sélection du mode lent

**POSITION 3:** Sélection du mode rapide

**Pour chaque modes :**

- **Faires le traitement des LAVES glaces avant et arrière en faisant clignoter .**
- **L'essuie-glace arrière si il est activé via un interrupteur à levier.**

- **Mode distant :**

Pour chaque position du comodo du volant DT-M001 récupérer les trames du bus can et les traiter avec des filtres

**POSITION 0:** Aucun mode sélectionnée donc faire le traitement que des laves glace AV/AR et éviter que les essuie glace AV et AR obstrue la vue de l'utilisateur.

**POSITION I :**Sélection du mode AUTOMATIQUE donc prendre en compte le capteur de pluie

MODE SÉLECTIONNE AUTOMATIQUE PAS DE PLUIE si le capteur de pluie est en position 0

MODE SÉLECTIONNE AUTOMATIQUE PLUIE FAIBLE si le capteur de pluie est en position 1

MODE SÉLECTIONNE AUTOMATIQUE PLUIE MOYENNE avec potentiomètre si le capteur de pluie est en position 2

MODE SÉLECTIONNÉ AUTOMATIQUE PLUIE FORTE si le capteur de pluie est en position 3

**POSITION 1:** Sélection du mode lent

**POSITION 2:** Sélection du mode rapide

### Pour chaque cas prendre en compte l'essuie glace arrière et les laves glaces AV/AR

- Gestion des moteurs et des temps d'intermittences pour les deux modes :

Pour simuler les essuie glace AV et AR nous utiliserons deux moteurs et deux fin de courses pour détecter un coup d'essuie glace. La vitesse des moteurs ne changera jamais il n'y a que le temps d'intermittences entre deux coup d'essuie glace qui changera !

Pour le mode lent et automatique pluie faible le temps d'intermittences est de 3 secondes pour les essuie-glaces avant.

Pour le mode rapide et automatique pluie forte le temps d'intermittences doit être de 0.5 seconde pour les essuie-glaces avant.

Pour le mode automatique pluie moyenne le temps d'intermittences doit être réglable par l'utilisateur à l'aide du potentiomètre pour les essuie-glaces avant.

Pour l'essuie-glace arrière sont temps d'intermittence sera fixes est sera de 3 secondes.

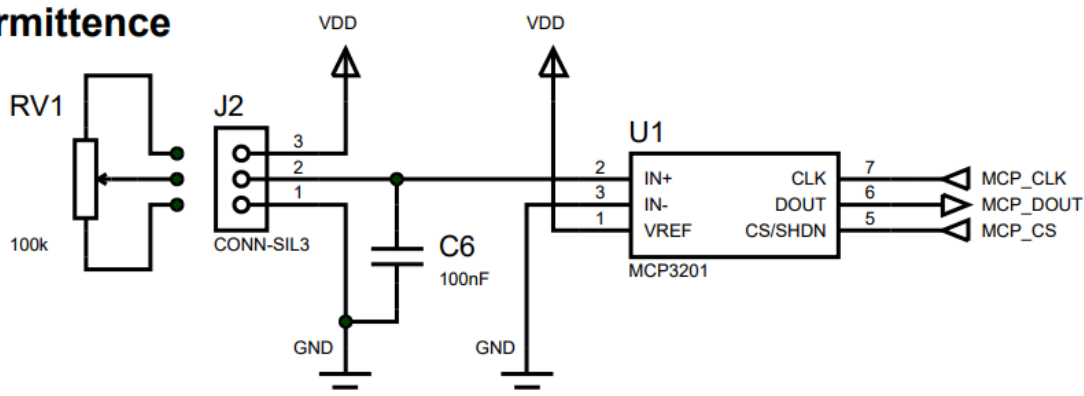
Utiliser la liaison RS232 pour débbugger et voir tous les modes !

### 3. Étude des cartes d'un point de vue électronique afin de déterminer les entrées et les sorties et réalisation la programmation des périphériques.

- Nous allons étudié le périphériques MCP3201 :

L'objectif de ce composant est de réaliser la fonction d'un convertisseur analogique numérique sur douze bits. Pour nous il nous permettra d'acquérir la valeur du potentiomètre saisie par l'utilisateur et ensuite la traiter pour gérer le temps d'intermittence en fonction du potentiomètre.

#### intermittence



Ils disposent de plusieurs broches avec leur rôles décrits par le tableau suivant :

Symbol	Description
$V_{REF}$	Reference Voltage Input
IN+	Positive Analog Input
IN-	Negative Analog Input
$V_{SS}$	Ground
$\overline{CS}/SHDN$	Chip Select/Shutdown Input
D <sub>OUT</sub>	Serial Data Out
CLK	Serial Clock
$V_{DD}$	+2.7V to 5.5V Power Supply

### 3.11 Entrée analogique positive (IN+)

Entrée analogique positive. Cette entrée peut varier de IN- à  $V_{REF} + IN-$ .

### 3.12 Entrée analogique négative (IN-)

Entrée analogique négative. Cette entrée peut varier de  $\pm 100$  mV par rapport à  $V_{SS}$ .

### 3.13 Sélection de puce / Mise hors tension (CS/SHDN)

La broche CS/SHDN est utilisée pour initier la communication avec le dispositif lorsqu'elle est tirée vers le haut et mettra fin à une conversion, plaçant le dispositif en mode veille à faible consommation lorsqu'elle est tirée vers le bas. La broche CS/SHDN doit être tirée vers le haut entre les conversions.

### 3.14 Horloge série (CLK)

La broche d'horloge SPI est utilisée pour initier une conversion et pour décaler chaque bit de la conversion au fur et à mesure de son déroulement. Consultez la section 6.2 "Maintenance de la vitesse minimale de l'horloge" pour les contraintes sur la vitesse de l'horloge.

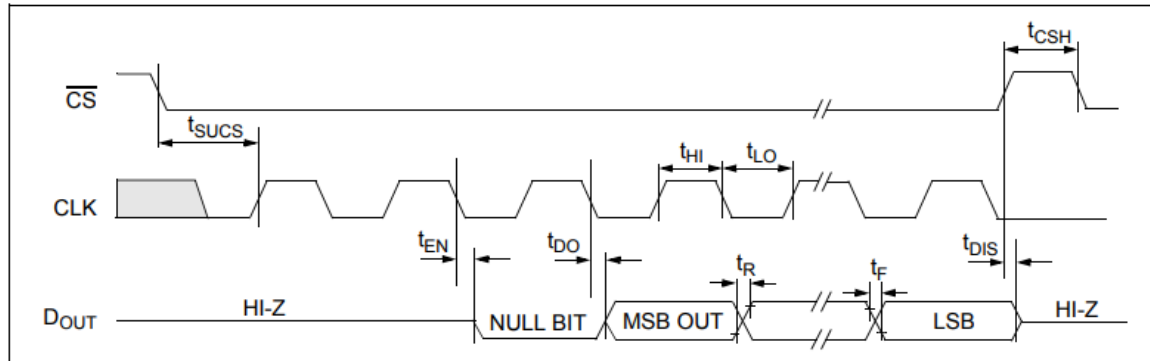
### 3.15 Sortie de données série (DOUT)

La broche de sortie de données série SPI est utilisée pour décaler les résultats de la conversion A/D. Les données changeront toujours sur le front descendant de chaque horloge au fur et à mesure de la conversion.

**Maintenant que nous avons étudié la partie électronique il faut étudier côté programmation :**

Voici une figure montrant le fonctionnement du composant MCP3201 l'idée est qu'il faut refaire le même fonctionnement c'est à dire dans un premier temps au début du projet mettre le chips select à 1 pour mettre le dispositif en veille ensuite le mettre à 0 pour initier la communication, quand on commence la conversion attendre le temps  $T_{suck}$  ensuite réaliser 3 coup d'horloges dans le vide attendre le temps  $t_{EN}$  puis réaliser 12 coup d'horloges de période  $T_{hi} + T_{lo}$  après faire un décalage vers la gauche sur 2 octet car c'est le MSB qui est en premier . Ensuite attendre le temps  $t_R$  et  $t_F$  pour la conversion qui sera directement pris par notre pic

car une instruction s'exécute en  $1/(5 \cdot 10^6) = 200$  ns. Après afficher le résultat et remettre à 1 le chip select.



**FIGURE 1-1:** Serial Timing.

Programme :

```
unsigned int16 read_MCP3201()
{
    int16 data=0;

    output_low(MCP_CS); // conversion lancer

    for (int i=0 ; i<16; i++){
        delay_cycles( 1 ); //312ns min (200ns_Tcycle)
        output_high(MCP_CLK);
        delay_cycles( 1 ); //312ns min (200ns_Tcycle)
        output_low(MCP_CLK);
        if(i>3){
            if (input(MCP_DOUT)) shift_left(&data,2,1); // décalage à gauche sur 2 octets (16
bits) avec 1L en entrée
            else shift_left(&data,2,0); // décalage à gauche sur 2 octets (16
bits) avec 0L en entrée

            delay_cycles( 1 );
        }
    }
    delay_cycles( 5 ); //tcsh 500ns
    output_high(MCP_CS); //arrêt
    return(data);
}
```

# RealTerm: Serial Capture Program 2.0.0.70

```

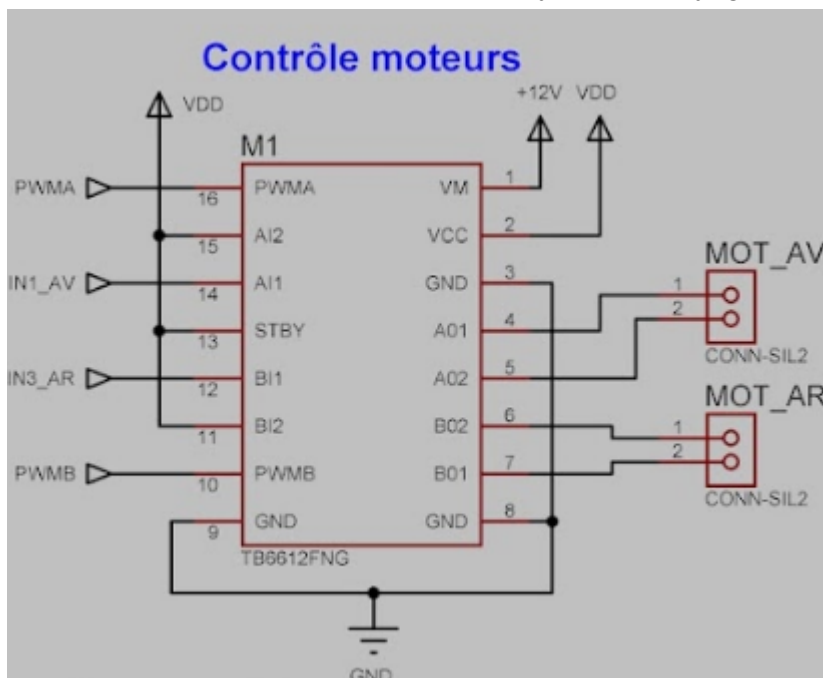
dÉbut du programme LFCR
Entrée dans le while true LFCR
Valeur du potentiomÉtre : 000E LFCR
Valeur du potentiomÉtre : 000E LFCR
Valeur du potentiomÉtre : 0251 LFCR
Valeur du potentiomÉtre : 026A LFCR
Valeur du potentiomÉtre : 05BB LFCR
Valeur du potentiomÉtre : 0D00 LFCR
Valeur du potentiomÉtre : 01F1 LFCR
Valeur du potentiomÉtre : 09DB LFCR
Valeur du potentiomÉtre : 00F5 LFCR
Valeur du potentiomÉtre : 00F1 LFCR
Valeur du potentiomÉtre : 091F LFCR
Valeur du potentiomÉtre : 0915 LFCR
Valeur du potentiomÉtre : 01A0 LFCR
Valeur du potentiomÉtre : 049F LFCR
Valeur du potentiomÉtre : 0DA0 LFCR
Valeur du potentiomÉtre : 0FF1 LFCR
Valeur du potentiomÉtre : 0FF5 LFCR
Valeur du potentiomÉtre : 0FF5 LFCR
Valeur du potentiomÉtre : 0FEE LFCR
Valeur du potentiomÉtre : 0FFB LFCR
Valeur du potentiomÉtre : 0FEE LFCR
Valeur du potentiomÉtre : 0FF5 LFCR
Valeur du potentiomÉtre : 0FF1 LFCR

```

Voici l'image que cela fonctionne.

## Nous allons étudier le périphériques TB6612 Sparkfun :

Le TB6612FNG est un circuit intégré de commande pour deux moteurs à courant continu. Il peut réaliser cinq modes tels que CW (sens horaire), CCW (sens anti-horaire), freinage court-circuit et mode arrêt, ainsi que la gestion de vitesse. Cependant, dans notre cas, nous n'utilisons que deux fonctions : sens anti-horaire et arrêt court, car pour un essuie-glace, nous avons besoin d'un seul sens de rotation et d'un arrêt/marche pour réaliser les intermittences entre chaque cycle de balayage.



Voici les rôles des pins et comment contrôler les moteur selon la datasheet du TB6612 Sparkfun :



**3.21 AI1 (Input ) et AI2 (Input ) :**

Ces broches contrôlent le sens de rotation du moteur A connecté au circuit. En inversant les niveaux logiques sur ces broches, vous pouvez faire tourner le moteur dans les deux sens.

**3.22 PWMA (input) :**

Cette broche est utilisée pour contrôler la vitesse du moteur A. En fournissant un signal PWM (modulation de largeur d'impulsion), vous pouvez ajuster la vitesse du moteur.

**3.23 STBY (Standby) :**

En mettant cette broche à un niveau logique bas (par exemple, en la reliant à la masse), vous désactivez le contrôleur de moteur, mettant le moteur en mode veille. Un niveau logique haut active le contrôleur.

**3.24 BI1 (Input ) et BI2 (Input) :**

Ces broches contrôlent le sens de rotation du moteur B connecté au circuit. Tout comme pour le moteur A, en inversant les niveaux logiques sur ces broches, vous pouvez faire tourner le moteur B dans les deux sens.

**3.25 PWMB :**

Cette broche est utilisée pour contrôler la vitesse du moteur B. Elle fonctionne de la même manière à PWMA.

**3.26 AO1 et AO2 (Outputs) :**

Ces broches sont connectées aux bornes du moteur A. La tension fournie à ces broches contrôle le fonctionnement du moteur A.

**3.27 BO1 et BO2 (Outputs) :**

Ces broches sont connectées aux bornes du moteur B. La tension fournie à ces broches contrôle le fonctionnement du moteur B.

**3.28 VCC :**

Cette broche est connectée à la source d'alimentation positive (typiquement entre 4,5 V et 13,5 V) pour alimenter le circuit.

**3.29 GND :**

Cette broche est connectée à la source d'alimentation négative (masse) et sert comme référence de tension.

**PS :** Nous ne ferons pas de pwm car la vitesse des essuie-glace ne dépend pas de la vitesse des moteurs mais dépend de la période entre chaque cycle de balayage. Donc on utilisera des niveaux logique pour la contrôler !

**Voici comment contrôler les moteurs selon les entrées :**



Input				Output		
IN1	IN2	PWM	STBY	OUT1	OUT2	Mode
H	H	H/L	H	L	L	Short brake
L	H	H	H	L	H	CCW
		L	H	L	L	Short brake
H	L	H	H	H	L	CW
		L	H	L	L	Short brake
L	L	H	H	OFF (High impedance)		Stop
H/L	H/L	H/L	L	OFF (High impedance)		Standby

Maintenant que nous avons étudié la partie électronique il faut étudier côté programmation :

```
#include <Prog_Moteur.h>
#include <Pin.h>

Void demarre_m_av(){//fonction demare le moteur avant
    output_low(IN1);
    output_high(PWM_EG_AV);
}

Void demarre_m_ar(){//fonction demare le moteur arrière
    output_low(IN3);
    output_high(PWM_EG_AR);
}

Void stop_m_av(){//fonction demare le moteur avant
    output_high(IN1);
    output_low(PWM_EG_AV);
}

Void stop_m_ar(){//fonction demare le moteur arrière
    output_high(IN3);
    output_low(PWM_EG_AR);
}

#INT_TIMER0
void TIMER0_isr(void)
{
}

#INT_TIMER1
void TIMER1_isr(void)
{
}

#INT_TIMER3
void TIMER3_isr(void)
```

```

{

}

#INT_EXT1
void EXT1_isr(void)
{

}

#define CAN_USE_EXTENDED_ID FALSE

#include <can-18F4580.c>

void main()
{
//SET TRIS
    Set_tris_a(0xFF);
    Set_tris_b(0b11011011);
    set_tris_c(0b10111111);
    set_tris_d(0x04);
    set_tris_e(0xFF);
//MOTEUR Eteint
    output_high(PWM_EG_AV);
    output_high(PWM_EG_AR);
    output_high(PIN_D7);
    output_high(PIN_D6);
//ADC
    output_high(MCP_CS); // arrêt de l'ADC externe
    output_low(MCP_CLK); // clk au NL0

    printf("début du programme \n\r");
//ADC
    setup_adc_ports(NO_ANALOGS, VSS_VDD); // Port A pas de analog
    setup_timer_0(RTCC_INTERNAL|RTCC_DIV_256); //3,4 s overflow
    setup_timer_1(T1_INTERNAL|T1_DIV_BY_8); //104 ms overflow
    setup_timer_3(T3_EXTERNAL | T3_DIV_BY_8); //104 ms overflow
    enable_interrupts(INT_TIMER0); // Active le timer 0
    enable_interrupts(INT_TIMER1); //Active le timer 1
    enable_interrupts(INT_TIMER3); //Active le timer 3
    ext_int_edge( 1, H_TO_L); // Front descendant sur IT externe 1
    enable_interrupts(INT_EXT1); // Active l'interruption Externe1
    disable_interrupts(GLOBAL); // désactiver les IT globale car nous commandons que les moteurs


//    can_init();

    while(TRUE)
    {
        demarre_m_av();
        demarre_m_ar();
    }
}

```

```
    delay_ms(3000);  
    stop_m_av();  
    stop_m_ar();  
    delay_ms(3000);  
}  
  
}
```

Les moteurs fonctionnent correctement :

 SAE\_Programme moteur.mp4

- **Maintenant nous allons étudier toutes les entrées/sorties de toutes les cartes et leurs rôles !**

Voici le tableau des entrées et des sorties :

Pin	E/S	Nom	Rôle
A0	E	EG_0	Position comodo essui-glace
A1	E	EG_1	Position comodo essui-glace
A2	E	CP_0	Capteur de pluie
A3	E	CP_1	Capteur de pluie
A4	E	LG_AV	Lave glace avant bp détecte appuis en 0L
A5	E	LG_AR	Lave glace arrière bp détecte appuis en 0L
A6	E	OSCILATEU	OSCILATEUR (FREQ 20Mhz) Le specifier dans le compilateur PCW (projet que c'est un quartz exterieur )
A7		R	
B0	E tout ou rien	INT0	Interruption non utilisée
B1	E tout ou rien	INT1	Interruption sur front descendant car ( OU LOGIQUE si il y a un front descendant il y a alors un des deux fc qui se sont activer )
B2	S	CAN_TX	Transmissions de can
B3	E	CAN_RX	Réception du can
B4	E	local/distance	Choix de mode
B5	S	LED_V	led verte pour dire que elle marche (interruption sur timer)
B6	E	PGC	Pick Flash programmation clock
B7	E et S	PGD	Pick Flash programmation data
C0	S	RC0	Rôle affichage I2C
C1	S	RC1	Rôle affichage I2C
C2	S	NC	non connecté
C3	S	SCL	Rôle affichage I2C
C4	E	SDA	Rôle affichage I2C
C5	S	RC5	Rôle affichage I2C
C6	S	TX_DEBUG	

C7	E	RX_DEBUG	
D0	S	MCP_CS	Nous permet de choisir quand le composant communique en 0L il communique et à 1L il ne peut plus communiquer quand on ne s'en sert pas on doit le mettre à 1L
D1	S	MCP_CLK	L'horloge du moteur
D2	E	MCP_DOUT	Résultat de la conversion analogique à numérique le convertisseur 12bit (4096) /// Objectif c'est de régler le potentiomètre en fonction de ce que l'on souhaite s'il y a des gros gouttes ou non .
D3	S	LED_D4	la led signifiant que le moteur est en fonctionnement
D4	S	PWM_EG_A V	Vitesse de moteur avant (0 ou 1)
D5	S	PWM_EG_A R	Vitesse de moteur arrière (0 ou 1)
D6	S	IN1	Sens de rotation du moteur avant
D7	S	IN3	Sens de rotation du moteur arrière
E0	E	Came_EG_A V	fin de course avant
E1	E	Came_EG_A R	fin de course arrière
E2	E	Com_EG_AR	pour activer l'essui glace arrière à 0L
E3	E	RESET_Prog	Réinitialiser le programme

Ensuite nous avons réalisé le filtre pour le CAN nous souhaitons que avoir l'id 0x094 qui correspond à la trame du commodo du volant :

BIT	15	14	13	12	11	10	9	8	5	4	3	2	1	0
ID : 0x094	0	0	0	0	0	0	1	0	0	1	0	1	0	0
MASQUE	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FILTRE	0	0	0	0	0	0	1	0	0	1	0	1	0	0

## 4. Programmation du projet.

D'abord nous devons calculer les différent endroit on nous ferons commencer notre Timer0

Formule:  $\frac{\text{Durée désirée}}{T_{\text{cycle}} * \text{Prescaleur}} - TMR1MAX = -N$

pour 3s:  $2^{16} + \frac{0,4}{(4/20*10^6)*256} = -57723 + 2^{16} = 7812$

pour 0.5S:  $2^{16} + \frac{2,9}{(4/20*10^6)*256} = -121306 + 2^{16} = 55770$

pour TIMER1 l'arrière :

100MS FIX :  $2^{16} + \frac{4*10^{-3}}{(4/20*10^6)*8} = -68036 + 2^{16} = 2500$

Dans un premiers temps nous avons réalisé la configuration hardware/software (PIN , FILTRE ,INTERRUPTION ,ADC ect ...) et ensuite nous avons fait une fonctions lecture qui va avoir le rôles de faire l'images des entrées de notre systemes :

```
Void lecture()
{
//LECTURE DES ENTREES
E_EG_0=input(EG_0);
E_EG_1=input(EG_1);
E_CP_0=input(CP_0);
E_CP_1=input(CP_1);
E_LG_AV=input(LG_AV);
E_LG_AR=input(LG_AR);
E_LOCAL_DISTANCE=input(LOCAL_DISTANCE);
E_CAME_EG_AV=input(CAME_EG_AV);
E_CAME_EG_AR=input(CAME_EG_AR);
E_COM_EG_AR=input(COM_EG_AR);
}
```

Après nous avons réaliser une fonction automate qui aura comme rôles de réaliser les traitement d'une variable état en fonction de c'est entrées :

Voici un tableau pour vous expliquer l'attribution de la variable état en fonction des entrées en mode locale :

E\_LOCAL\_DISTANCE est l'entrée qui sélectionne le mode locale (0 logique ) ou distant (1 logique)

E_LOCAL_DISTANCE	E_EG_0	E_EG_1	E_CP_0	E_CP_1	ETAT	Mode	Intermittence
0	0	0	X	X	0	Aucun mode sélectionné	X
0	1	0	0	0	1	MODE AUTOMATIQUE sans pluie	X
0	0	1	X	X	2	Mode sélectionné lent	3 secondes
0	1	1	X	X	3	Mode sélectionné rapide	0.5secondes
0	1	0	1	0	4	Mode sélectionné automatique pluie faible	3 secondes

0	1	0	0	1	5	Mode sélectionné automatique pluie moyenne donc selon la valeur du potentiomètre	3 secondes
0	1	0	1	1	6	Mode sélectionné automatique pluie forte	0.5secondes
1	X	X	X	X	X	Mode distant donc en fonction de la trames et du capteurs de pluie	X

### Pour le mode distant avec les trames du volant:

ID	ETAT	TRAME ES AV	TRAME ES AV et AR	TRAME ES AV et LG AV	TRAME ES AV et LG AR	TRAME ES AV et AR et LG AV
0x094	0 Aucun mode sélectionnée	20 00 02 00 00 4A	20 04 02 00 00 4A	20 08 02 00 00 4A	20 02 02 00 00 4A	20 0C 02 00 00 4A
0x094	1 MODE SÉLECTIONNE AUTOMATIQUE	20 20 02 00 00 4A	20 24 02 00 00 4A	20 28 02 00 00 4A	20 22 02 00 00 4A	20 2C 02 00 00 4A
0x094	2 MODE SÉLECTIONNÉ LENT	20 40 02 00 00 4A	20 44 02 00 00 4A	20 48 02 00 00 4A	20 42 02 00 00 4A	20 4C 02 00 00 4A
0x094	3 MODE SÉLECTIONNÉ RAPIDE	20 80 02 00 00 4A	20 84 02 00 00 4A	20 88 02 00 00 4A	20 82 02 00 00 4A	20 8C 02 00 00 4A

L'identifiant (ID) requis pour la trame est 0x094, ce qui nous permettra de déterminer la position du commodo d'essuie-glace. À chaque état, on observe qu'un octet spécifique subit des modifications. Les bits de 0 à 3 indiquent la destination (quel moteur d'essuie-glace, quel lave-glace), tandis que les bits de 4 à 7 nous renseignent sur l'état actuel. Les états 4 à 6 précédemment évoqués ne sont pas répertoriés dans ce tableau, car ils correspondent à la même trame que l'état 1.

L'état 1 mode automatique pluie faible : Trame du mode automatique et des capteur de pluie en position 00

L'état 4 mode automatique pluie faible : Trame du mode automatique et des capteur de pluie en position 01

L'état 5 mode automatique pluie moyenne : Trame du mode automatique et des capteur de pluie en position 10

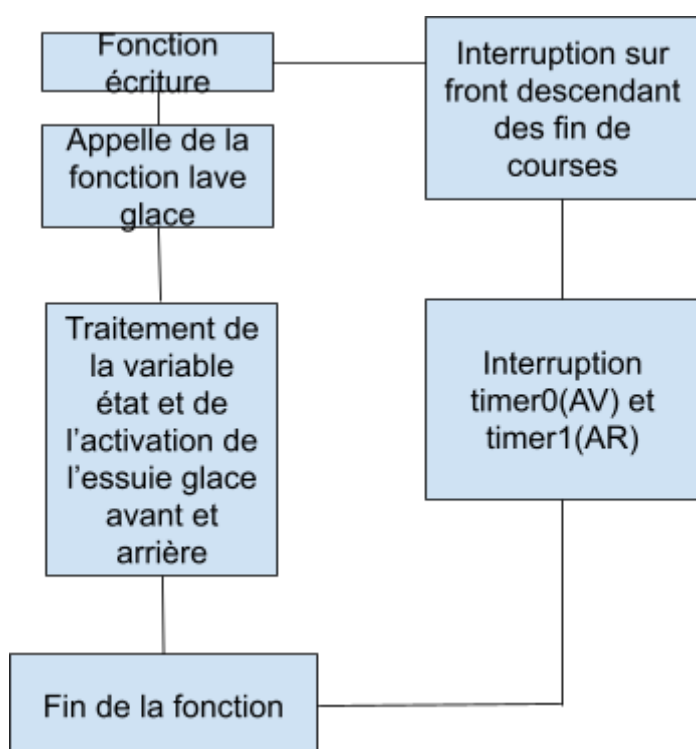
L'état 6 mode automatique pluie forte : Trame du mode automatique et des capteur de pluie en position 11

### Le programme de la fonction automate se trouve dans l'annexe :

Qui sera composé du traitement des entrées et des trames qui permettent d'affecter une valeur à une variable état qui sera utilisé dans la fonction écriture pour réaliser le traitement des sorties! Nous afficherons sur la RS232 et le LCD l'état ou on est . Les flags action sont utilisés pour éviter que des actions se répètent inutilement, par exemple afficher sur la RS232 l'état ou on est constamment . Avec les flags action on affiche que une fois l'état ou on est et si les entrées venait à changer alors on affiche le nouvel état .

Après avoir effectué des tests, cette fonction marche .





Ensuite nous avons réalisé la fonction écriture qui s'occupera de traiter les sorties en fonctions de la variable etats et des bp lave glace et des trames des laves glaces .

Nous utiliserons les timer1 et timer2 pour réaliser les temps entres chaque balayage

Nous utiliserons une IT pour détecter le changement de fronts sur les fin de courses celà nous servira à détecter quand nous avons fait 1 coup d'essuie-glace.

### **Interruption sur front descendant sur l'un des deux fin de courses :**

**Début IT** il y a changement sur front descendant sur l'un des deux fin de courses

Lire les entrées des deux fin de courses

**FIN IT**

### **Fonction lave\_glace()**

Si (Le boutons du lave glace avant est pressé et on est en locale ou que il y a une trame correspondant au lave glace avant) Alors

Afficher "Lave glace avant"

Fin Si

Si (Le boutons du lave glace arrière est pressé et on est en locale ou que il y a une trame correspondant au lave glace arrière) Alors

Afficher "Lave glace arrière"

Fin Si

## Fin Fonction

**Cet algorithme a pour but d'éviter d'obstruer la vision de l'utilisateur quand nous avons aucun mode sélectionné ou que l'essuie-glace arrière n'est pas activé .Nous prendrons toujours en compte les IT sur le fin de course.**

SI (L'ESSUIE GLACE ARRIERE N'EST PAS ACTIVER ET ETAT EST DIFFÉRENT DE 1) OU ETAT == 0  
ALORS

SI LE FIN DE COURSE ARRIERE N'EST PAS ACTIVER ALORS

demarre\_m\_ar()

flg\_d\_timer\_1 = 0

SINON SI LE FIN DE COURSE ARRIÈRE EST ACTIVÉ ALORS

stop\_m\_ar()

flg\_d\_timer\_1 = 0

FIN SI

FIN SI

SI ETAT == 0 ALORS

SI LE FIN DE COURSE AVANT N'EST PAS ACTIVER ALORS

demarre\_m\_av()

flg\_d\_timer\_0 = 0

SINON SI LE FIN DE COURSE AVANT EST ACTIVÉ ALORS

stop\_m\_av()

flg\_d\_timer\_0 = 0

FIN SI

FIN SI

**Cet algorithme a pour but de réaliser les fonction de l'essuie-glace avant avec les IT sur les deux timer et l'it sur le fin de course.**

**SI ETAT == 6 OU ÉTAT == 3 ALORS**

SI LE FIN DE COURSE AVANT N'EST PAS ACTIVÉ ALORS

demarre\_m\_av()

flg\_d\_timer\_0 = 0

SINON

SI flg\_d\_timer\_0 == 0 ALORS

flg\_d\_timer\_0 = 1

On met le timer0 a 2,9Seconde pour avoir un temps de 0,5S

stop\_m\_av()

FIN SI

FIN SI

**FIN SI**

**IT sur timer0 overflow à 3,4S :**

SI ETAT == 3 OU ETAT == 6 ALORS

demarre\_m\_av() // démarrer le moteur avant

**FIN SI**

Cet algorithme sera répété plusieurs fois avec quelques modifications, par exemple le temps du timer pour les différents modes .

**Voici la gestion de l'essuie glace arrière :**

**SI (ETAT == 2 OU ETAT == 3 OU ETAT == 4 OU ETAT == 5 OU ETAT == 6) ALORS**

```

SI l'essuie glace arrière est activer ALORS
  SI le fin de course est activer ALORS
    démarre_m_ar() // démarrer le moteur arrière
    flg_d_timer_1 = 0
  SINON
    SI flg_d_timer_1 == 0 ALORS
      flg_d_timer_1 = 1
      mettre le timer à 4ms
      overflow = 0
      stop_m_ar() // arrêter le moteur arrière
    FIN SI
  FIN SI
FIN SI

```

**FIN SI**

**IT SUR OVERFLOW DU TIMER 0 Tous les 104MS**

```

SI L'ESSUIE GLACE ARRIERE EST ACTIF OU ETAT == 1 ALORS
  SI ETAT == 2 OU ETAT == 1 OU ETAT == 3 OU ETAT == 4 OU ETAT == 5 OU ETAT == 6 ALORS
    SI overflow > 29 ALORS
      overflow = 0
      démarre_m_ar() // démarrer le moteur arrière
    FIN SI
    incrementer overflow
  FIN SI
FIN SI
METTRE LE TIMER1 A à 0,004S //
FIN DE L'IT

```

**Ensuite nous avons réalisé l'affichage sur le lcd à l'aide d'une fonction et sa bibliothèque :**

```

LCD_AFF() // Fonction qui affiche sur le LCD
Début
  Si affiche_periode > 4 Alors // Si on est à 500ms, alors afficher
    GROVE_LCD_clear() // Effacer le LCD

  Si E_LOCAL_DISTANCE égal à 0 Alors // Si on est en mode local
    Mettre le curseur à la colonne 0 et ligne 0
    Afficher sur le LCD "L" // AFFICHER L POUR MODE LOCALE
  Sinon // Sinon
    Mettre le curseur à la colonne 0 et ligne 0
    Afficher sur le LCD "D" // AFFICHER D POUR MODE DISTANT
  FinSi

  Selon ETAT // Cas parmi la variable ÉTAT

```

Cas 0 : // Cas 0, mode aucun

Mettre le curseur à la colonne 2 et ligne 0

Afficher sur le LCD "AUCUN"

FinCas

Cas 1 : // Cas 1, mode automatique

Mettre le curseur à la colonne 2 et ligne 0

Afficher sur le LCD "AUTO"

FinCas

Cas 2 : // Cas 2, mode lent

Mettre le curseur à la colonne 2 et ligne 0

Afficher sur le LCD "LENT"

FinCas

Cas 3 : // Cas 3, mode rapide

Mettre le curseur à la colonne 2 et ligne 0

Afficher sur le LCD "RAPIDE"

FinCas

Cas 4 : // Cas 4, mode auto pluie faible

Mettre le curseur à la colonne 2 et ligne 0

Afficher sur le LCD "AUTO FAIBLE"

FinCas

Cas 5 : // Cas 5, mode auto moyen

Mettre le curseur à la colonne 2 et ligne 0

Afficher sur le LCD "AUTO MOYEN"

FinCas

Cas 6 : // Cas 6, mode auto pluie forte

Mettre le curseur à la colonne 2 et ligne 0

Afficher sur le LCD "AUTO FORT"

FinCas

Fin Selon

affiche\_periode = 0 // Réinitialiser le compteur de période d'affichage

FinSi

Fin

**Conclusion : Ce projet nous aura permis d'étudier le bus can en intégralité . Il nous aura permis d'apprendre de nous mêmes et de construire des algorithmes efficaces afin de satisfaire le cahier des charges . Il nous appris la rigueur et la discipline que doit imposer un projet de cette envergure.**

# ANNEXES

```

/*****
*SAE ESE Essuie glace avec communication bus CAN      date : 11/01/2024
*****/
*Gestion d'un mode Local ou distant
*Gestion et Traitement de La variable d'état
*Faire un affichage de L'état présent avec RS232 et LCD
*Fonction Lecture qui permet de recevoir toute Les entrées
*Fonction Automate qui attribue la valeur etat en fonction des entrées ou des trames
*Fonction Ecriture qui en fonction de la variable état vas traité Les sorties (moteur,led)
*Timer1 qui permet Le clignotement de la LED des lave-glaces et gestion essuie glace arrière
*Timer0 gestion de L'essuie glace avant
*Les Timers sont utilisée pour gere Les intermittences entre chaque balayages
*****/

#include <SAE_ESSUIE_GLACE.h>//point H configuration matérielle
#include <Pin.h>//point H configuration des pins
#include <LCD_GROVE.h> // Ajoue de la librairie LCD_GROOVE

/*****
CAN
*****/
#define CAN_USE_EXTENDED_ID FALSE // CAN2.0 A (ID sur 11 bits)
#define CAN_DO_DEBUG FALSE // mode DEBUG (RS232)=> CAN_DO_DEBUG = TRUE
#define CAN_BRG_PRESCALAR 4 // BRP -->4 Tq = [2 x (BRP+1)]/ Clock 125kbit/s
#define CAN_BRG_SEG_2_PHASE_TS TRUE // TRUE --> choix libre de TSEG2
#define CAN_BRG_SAM 0 // un seul echantillonage du bit
#define CAN_BRG_SYNCH_JUMP_WIDTH 0 // Synchronized Jump Width bit = 0 => SJW = 1 Tq
#define CAN_BRG_PHASE_SEGMENT_1 6 // Phase Segment 1 = 6 => TSEG1 = 7 Tq
#define CAN_BRG_PHASE_SEGMENT_2 3 // Phase Segment 2 = 3 => TSEG2 = 4 Tq
#define CAN_BRG_PROPAGATION_TIME 3 // Propagation Time =3 => Tprop = 4 Tq

#include <can-18F4580.c>

int32 rx_id;          // Id du message reçu
int8 i,               // indice de boucle
    rx_len,           // nb octets de data (trame CAN)
    buffer[8];        // max 8 octets de data
struct rx_stat rxstat; // structure rxstat de type rx_stat (voir fichier can-18f4580.h)
int8 data36[8]={0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00}; //Envoyer la trame pour faire marché le volant (id
0x36 )
int PERIODE_CAN=0; //Envoyer tous les 100ms l'id 36
int FLAG_CAN_IT=0; //Pour prévenir que nous avons reçue une trame can
/*****
Variable pour lire les entrées de notre systèmes
*****/
int E_EG_0=0;int E_EG_1=0;int E_CP_0=0;int E_CP_1=0;int E_LG_AV=0;int E_LG_AR=0;int E_LOCAL_DISTANCE=0;int
E_CAME_EG_AV=0;int E_CAME_EG_AR=0;int E_COM_EG_AR=0; //Pour lire l'état de nos entrées

int FLG_ACTION_1=0;int FLG_ACTION_ANCIEN=1000;int FLG_ACTION_LOCALE=0; // Pour eviter que on printf a l'infinie
dans le moniteur serie
int FLG_ACTION_ANCIEN_2=999; // pour eviter d'afficher tous le temps sur le RS232
int ETAT=0; // Variable etat pour definir :
/*
Si etat = 0 AUCUN MODE SELECTIONNE
Si etat = 1 MODE SELECTIONNE AUTOMATIQUE donc que l'arrière

```

```

Si etat = 2 MODE SELECTIONNE LENT
Si etat = 3 MODE SELECTIONNE RAPIDE
Si etat = 4 MODE SELECTIONNE AUTOMATIQUE_PLUIE_FAIBLE
Si etat = 5 MODE SELECTIONNE AUTOMATIQUE_PLUIE_MOYENNE avec potentiomettre
Si etat = 6 MODE SELECTIONNE AUTOMATIQUE_PLUIE_FORTE
*/

int affiche_periode=0;//pour afficher tous les 300ms sur le LCD
int LG_AV_TIMER=0;//POUR LA POMPE DU TIMER DU LG AV
int LG_AR_TIMER=0;//POUR LA POMPE DU TIMER DU LG AR

int flg_d_timer_0=0;// flg qui permet de démarer le timer 0 à une certains temps pour l'essuie glace avant
int flg_d_timer_1=0;// flg qui permet de démarer le timer 1 à une certains temps pour l'essuie glace arrière
int overflow=0;// pour le timer 1 car nous avons un overflow tous les 100ms cependant nous souhaitons que notre
essuie glace arrière est une periode de 3s entre chaque balayage arrière
int clignotement=0;// Pour faire clignoter la led pour simuler une pompe pour les LAVE GLASSE

Void demarre_m_av(){//fonction demare le moteur avant
    output_low(IN1);
    output_high(PWM_EG_AV);
}

Void demarre_m_ar(){//fonction demare le moteur arrière
    output_low(IN3);
    output_high(PWM_EG_AR);
}

Void stop_m_av(){//fonction demare le moteur avant
    output_high(IN1);
    output_low(PWM_EG_AV);
}

Void stop_m_ar(){//fonction demare le moteur arrière
    output_high(IN3);
    output_low(PWM_EG_AR);
}

Void lave_glace(){//fonction lave glace
if(!E_LG_AV|buffer[1]==0x08|buffer[1]==0x28|buffer[1]==0x48|buffer[1]==0x88|buffer[1]==0x0C|buffer[1]==0x
2C|buffer[1]==0x4C|buffer[1]==0x8C|buffer[1]==0x0A|buffer[1]==0x2A|buffer[1]==0x4A|buffer[1]==0x8A){//si
il y a une trame avec le lave glace ou que le bp lg est presse
    printf("Lave glace avant \n \r");//affiche lave glace avant
    LG_AV_TIMER=1;//LG AV clignotement led pour le timer quand il est detecte
}
else{LG_AV_TIMER=0;}//LG AV désactiver
if(!E_LG_AR|buffer[1]==0x02|buffer[1]==0x22|buffer[1]==0x42|buffer[1]==0x82|buffer[1]==0x0A|buffer[1]==0x
2A|buffer[1]==0x4A|buffer[1]==0x8A){//si le bp lg arrière et presse alors
printf("Lave glace arrière \n \r");
LG_AR_TIMER=1;                //LG AR clignotement led pour le timer quand il est detecte
}
else{LG_AR_TIMER=0;}//LG AR désactiver

    if(LG_AR_TIMER==1&&LG_AV_TIMER==1)//Si les 2 lave glace sont active
    {
        output_low(LED_V);                //allumer la LED
    }
}

```

```

else { //sinon
if(LG_AR_TIMER==1||LG_AV_TIMER==1){ //Si l'un des deux lave glace est activé alors
    if(clignotement>=5){ //Si la variable clignotement est égale ou supérieur a 5
        output_high(LED_V); //éteindre la LED
    } //Sinon
    else{output_low(LED_V);} //allumer la LED
}
else{output_high(LED_V);} //Sinon éteindre la LED verte
}
}

/*****
 * IT bus CAN message invalide
 *****/
#int_CANIRX
void CANIRX_isr(void)
{printf("\n\r INVALID MESSAGE");
}

/*****
 * IT bus CAN erreur
 *****/
#int_CANERR
void CANERR_isr(void)
{
//disable_interrupts(GLOBAL);
// plus d'IT autorisée
//! printf("\n\r CAN BUS ERR : COMSTAT %2LX\n\r ",REG_COMSTAT);
//! printf("\n\r RXERR = %u",REG_RXERRCNT);
//! printf("\n\r TXERR = %u",REG_TXERRCNT);
comstat.rx0ovfl=0;
comstat.rx1ovfl=0;
//! printf("\n\r STOP");
enable_interrupts(GLOBAL);
}

/*****
 * IT bus CAN ; le Buffer Réception 1 contient un message
 *****/
#int_CANRX1
void CANRX1_isr(void)
{
FLAG_CAN_IT=1; // flag de détection d'une IT activé
}

/*****
 * IT bus CAN ; le Buffer Réception 0 contient un message
 *****/
#int_CANRX0
void CANRX0_isr(void)
{
FLAG_CAN_IT=1; // flag de détection d'une IT activé
}

#INT_TIMER0
void TIMER0_isr(void) // Pour le moteur avant sachant que le timer0 possède un overflow au bout de 3.4s
{
if(ETAT==2||ETAT==3||ETAT==4||ETAT==5||ETAT==6)// si on est à l'état 2,3,4,5,6 le moteur avant marche
{
demarre_m_av();//démarré le moteur avant
}
}

```



```

}

#INT_TIMER1
void TIMER1_isr(void) //on utilise la variable overflow pour avoir 3seconde entre chaque ballaiage arrière
car notre timer à un overflow max de 104ms
    //et le set timer nous fait commence a 4ms donc on a une IT tous les 100MS comme ca on
    peut faire des ballaiage a un temps bien précis et envoyer la trame pour faire marcher les comodors
{
    PERIODE_CAN=0; // periode_can =0
    affiche_periode++; //Pour afficher sur le lcd à une periode de 300ms on incrémente tous les 100ms
    if(LG_AV_TIMER|LG_AR_TIMER){ //Si un des 2 flags est active alors on augmente le clignotement
        clignotement++; //incréméntation du clignotement
    }
    if(LG_AV_TIMER==0&&LG_AR_TIMER==0){ //Si aucun des 2 flags est active
        clignotement=0; //la variable clignotement est mis à 0
    }
    if(clignotement==10){clignotement=0;} //Si la variable clignotement a attint 10 on la remet a 0

    if(E_COM_EG_AR==0|ETAT==1)//SI l'essuie glasse arrière est activer alors
    {
        if(ETAT==2|ETAT==1|ETAT==3|ETAT==4|ETAT==5|ETAT==6){ //Si on est dans un des état citer
            if(overflow>29){ //Si la variable overflow est supérieur
a 29
                overflow=0; //on remet la variable overflow a 0
                demarre_m_ar(); //demare moteur arrière
            }
            overflow++; //la variable overflow augmente
        }
        Set_timer1(2500); //commence a 0,4Seconde
    }
}

#INT_TIMER3
void TIMER3_isr(void)
{
    //printf("TIMER 3\n\r");
    //set_timer3(2500); //On met le timer à 4ms
}

#INT_EXT1
void EXT1_isr(void)
{
    E_CAME_EG_AV=input(Came_EG_AV); //La variable E_CAME_EG_AV prend la valeur de la variable Came_EG_AV
    E_CAME_EG_AR=input(Came_EG_AR); //La variable E_CAME_EG_AR prend la valeur de la variable Came_EG_AR
}

/*****
* Filtrage pour ID 0x094 *
*****/

Void can_filtre()
{
    can_set_mode(CAN_OP_CONFIG); // D'abord passer en mode config
    can_set_id(RX0MASK,0x0FFF,FALSE); // Mask pour RXB0
    can_set_id(RX0FILTER0,0x094,FALSE); // Filtre n°0 associé à RBX0
    can_set_mode(CAN_OP_NORMAL); // Retour en mode opérationnel
    can_enable_filter(RXF0EN); // activation des filtres configurés précédemment
}

```

```
// #define MCP_CS PIN_D0 #define MCP_CLK PIN_D1 #define MCP_DOUT PIN_D2
unsigned int16 read_MCP3201()
{
    int16 data=0;//data reniasilisation de ça valeur

    output_low(MCP_CS);// conversion lancer

    for (int i=0 ; i<16; i++){
        delay_cycles( 1 );          //312ns min (200ns_Tcycle)
        output_high(MCP_CLK); //haut de la clock
        delay_cycles( 1 );          //312ns min (200ns_Tcycle)
        output_low(MCP_CLK); //etat bas de la clk pour genere une cclk
        if(i>3){
            if (input(MCP_DOUT)) shift_left(&data,2,1); // décalage à gauche sur 2 octets (16 bits) avec 1L en entrée
            else shift_left(&data,2,0); // décalage à gauche sur 2 octets (16 bits) avec 0L en entrée

            delay_cycles( 2 );
        }
    }
    delay_cycles( 6 ); //tcsh 500ns
    output_high(MCP_CS);//arrêt
    return(data);
}

/*
Si etat = 0 AUCUN MODE SELECTIONNE
Si etat = 1 MODE SELECTIONNE AUTOMATIQUE donc que l'arrière
Si etat = 2 MODE SELECTIONNE LENT
Si etat = 3 MODE SELECTIONNE RAPIDE
Si etat = 4 MODE SELECTIONNE AUTOMATIQUE_PLUIE_FAIBLE
Si etat = 5 MODE SELECTIONNE AUTOMATIQUE_PLUIE_MOYENNE avec potentiomettre
Si etat = 6 MODE SELECTIONNE AUTOMATIQUE_PLUIE_FORTE
*/

Void LCD_AFF();//fonction qui affiche sur le lcd
{
    if(affiche_periode>4){//Si on est à 500ms
        GROVE_LCD_clear();//clear le lcd
        if(E_LOCAL_DISTANCE==0)//Si on est en mode locale
        {
            GROVE_LCD_gotoxy(0,0);
            printf(GROVE_LCD_printf,"L");//AFFICHE L POUR MODE LOCALE

        }
        else//Sinon
        {
            GROVE_LCD_gotoxy(0,0);
            printf(GROVE_LCD_printf,"D");// AFFICHE D POUR MODE DISTANT

        }
    }
    Switch(ETAT)//CAS PARMIS LA VARIABLE ETAT
    {
        case 0:// CAS PARMIS 0 MODE AUCUN
            GROVE_LCD_gotoxy(2,0);
            printf(GROVE_LCD_printf, "AUCUN");
            break;
        case 1:// MODE AUTOMATIQUE
            GROVE_LCD_gotoxy(2,0);
    }
}
```

```

        printf(GROVE_LCD_printf, "AUTO");
        break;
    case 2://MODE LENT
        GROVE_LCD_gotoxy(2,0);
        printf(GROVE_LCD_printf, "LENT");
        break;
    case 3:// MODE RAPIDE
        GROVE_LCD_gotoxy(2,0);
        printf(GROVE_LCD_printf, "RAPIDE");
        break;
    case 4:// MODE AUTO PLUIE FAIBLE
        GROVE_LCD_gotoxy(2,0);
        printf(GROVE_LCD_printf, "AUTO FAIBLE");
        break;
    case 5:// MODE AUTO MOYEN
        GROVE_LCD_gotoxy(2,0);
        printf(GROVE_LCD_printf, "AUTO MOYEN");
        break;
    case 6:// MODE AUTO PLUIE FORTE
        GROVE_LCD_gotoxy(2,0);
        printf(GROVE_LCD_printf, "AUTO FORT");
        break;
    }
    affiche_periode=0;
}
}

```

```

Void lecture()
{
//LECTURE DES ENTREES
E_LOCAL_DISTANCE=input(LOCAL_DISTANCE);
E_EG_0=input(EG_0);
E_EG_1=input(EG_1);
E_CP_0=input(CP_0);
E_CP_1=input(CP_1);
E_LG_AV=input(LG_AV);
E_LG_AR=input(LG_AR);
E_CAME_EG_AV=input(CAME_EG_AV);
E_CAME_EG_AR=input(CAME_EG_AR);
E_COM_EG_AR=input(COM_EG_AR);

}

```

```

Void automate() //TRAITEMENT DES ENTREES
{
    if(E_LOCAL_DISTANCE==0){
        if(FLG_ACTION_LOCALE==0){printf("Mode locale");FLG_ACTION_LOCALE=1;}
        if(E_EG_0==0 && E_EG_1==0 )//AUCUN MODE
        {
            ETAT=0;
            FLG_ACTION_1=0;
            if(FLG_ACTION_1!=FLG_ACTION_ANCIEN){printf("Aucun mode sélectionne \n \r");}
        }
        if(E_EG_0==1 && E_EG_1==0 && E_CP_0==0 && E_CP_1==0)//Mode Automatique sans pluie donc potentiometre
        {
            ETAT=1;

```

```

        FLG_ACTION_1=1;
        if(FLG_ACTION_1!=FLG_ACTION_ANCIEN){printf("MODE AUTOMATIQUE DONC que l'essuie glasse arrière \n \r");}
    }
if( E_EG_0==0 && E_EG_1==1 )//Mode lent
{
    ETAT=2;
    FLG_ACTION_1=2;
    if(FLG_ACTION_1!=FLG_ACTION_ANCIEN){printf("Mode sélectionné lent \n \r");}
}
if(E_EG_0==1 && E_EG_1==1 )//Mode rapide
{
    ETAT=3;
    FLG_ACTION_1=3;
    if(FLG_ACTION_1!=FLG_ACTION_ANCIEN){printf("Mode sélectionné rapide \n \r");}
}
if(E_EG_0==1 && E_EG_1==0 && E_CP_0==1 && E_CP_1==0)// Mode Automatique pluie faible
{
    ETAT=4;
    FLG_ACTION_1=4;
    if(FLG_ACTION_1!=FLG_ACTION_ANCIEN){printf("Mode sélectionné automatique pluie faible \n \r");}
}
if(E_EG_0==1 && E_EG_1==0 && E_CP_0==0 && E_CP_1==1)//Mode Automatique pluie moyenne
{
    ETAT=5;
    FLG_ACTION_1=5;
    if(FLG_ACTION_1!=FLG_ACTION_ANCIEN){printf("Mode sélectionné automatique pluie moyenne donc
potentiometre \n \r");}
}
if(E_EG_0==1 && E_EG_1==0 && E_CP_0==1 && E_CP_1==1)//Mode Automatique pluie forte
{
    ETAT=6;
    FLG_ACTION_1=6;
    if(FLG_ACTION_1!=FLG_ACTION_ANCIEN){printf("Mode sélectionné automatique pluie forte \n \r");}
}
    FLG_ACTION_ANCIEN=FLG_ACTION_1;
}
else{
    if(FLG_ACTION_LOCALE==1){printf("Mode distant");FLG_ACTION_LOCALE=0;}

    if(PERIODE_CAN==0)// si PERIODE CAN = 0 pour envoyer periodiquement la trame de donnés pour faire
marcher les commandos
    {
        can_putd(0x36,data36, 8 , 1, FALSE, FALSE);// trame de données pour faire marche les comodos
        PERIODE_CAN=1;// Pour eviter de retourne en boucle de dans . INTERRUPTION TIMER 2
    }
    if (FLAG_CAN_IT) // SI trame reçue (plusieurs registres possibles : RX0 ou
RX1)
    {
        FLAG_CAN_IT=0; // ALORS RAZ flag
        printf("\n\rCOMSTAT = %2LX\n\r",REG_COMSTAT);
        if (can_getd(rx_id,&buffer[0],rx_len,rxstat)) // SI la trame contient des données (can_getd
renvoie 1 dans ce cas)
        {printf("RX_ID %LX : (%u) ",rx_id,rx_len); // ALORS afficher ID reçu et nb de data
            if (!rxstat.rtr) // SI trame de data (RTR=0)
            { // Alors
                for (i=0; i<rx_len;i++) // POUR les data reçues
                    printf("%X ",buffer[i]); // afficher la data
            } // Fin de POUR
        } // Fin SI
    }
}

```

```

    if (rxstat.rtr) printf(" RTR=1 ");          //          SI trame de requête (RTR=1) ALORS
afficher 'RTR=1' Fin SI
    if (rxstat.err_ovfl) printf(" OVFL ");      //          SI ALORS afficher 'OVFL' Fin SI
    if (comstat.ewarn) printf("CNT>96");
    printf("\n\r");                          //          sauter une ligne
}                                              //          Fin SI
} //      Fin SI
if( buffer[1]==0x00 || buffer[1]==0x04 || buffer[1]==0x02 || buffer[1]==0x08 || buffer[1]==0x0C
||buffer[1]==0x0A )// ETAT 0 avec tous c'est possibilités donc mode selectionner aucun
{
    ETAT=0;//aucun mode
}
if( (buffer[1]==0x20 || buffer[1]==0x24 || buffer[1]==0x22 || buffer[1]==0x28 || buffer[1]==0x2C
||buffer[1]==0x2A))// ETAT 1 avec tous c'est possibilités donc mode selectionner automatique sans pluie
{
    if(E_CP_0==0 && E_CP_1==0){
        ETAT=1;//mode automatique sans pluie donc que l'essuie glasse arrière
    }
    if(E_CP_0==1 && E_CP_1==0){// mode automatique pluie faible
        ETAT=4;//pluie faible
    }
    if( E_CP_0==0 && E_CP_1==1){//mode automatique pluie moyenne
        ETAT=5;
    }
    if(E_CP_0==1 && E_CP_1==1){// mode automatique pluie forte
        ETAT=6;//pluie forte
    }
}
if( (buffer[1]==0x40 || buffer[1]==0x44 || buffer[1]==0x42 || buffer[1]==0x48 || buffer[1]==0x4C
||buffer[1]==0x4A))// ETAT 2 avec tous c'est possibilités donc mode selectionner lent
{
    ETAT=2;//mode lent
}
if( (buffer[1]==0x80 || buffer[1]==0x84 || buffer[1]==0x82 || buffer[1]==0x88 || buffer[1]==0x8C
||buffer[1]==0x8A))// ETAT 3 avec tous c'est possibilités donc mode selectionner rapide
{
    ETAT=3;//mode rapide
}
if((buffer[1]==0x04 || buffer[1]==0x84 || buffer[1]==0x24 || buffer[1]==0x44 || buffer[1]==0x0C
||buffer[1]==0x2C || buffer[1]==0x4C ||buffer[1]==0x8C))//Si on a cette trame on active l'essuie glace arrière
{
    E_COM_EG_AR=0;//on active l'essuie glace arrière
}
else//sinon
{
    E_COM_EG_AR=1;//on desactive l'essuie glace arrière
}
}
LCD_AFF();          //Débugage et affichage sur LCD
}

```

```

Void ecriture()
{
    LAVE_GLACE();//APPELLE DE LA FONCTION LAVE GLACE
    if(E_COM_EG_AR==1|ETAT==0)//SI LE MODE ESSUIE GLASSE N'EST PAS ACTIF ou que on n'est pas a 1 etat 2 ALORS
    EVITER D'OBSTURER LA VUE
    {
        if(E_CAME_EG_AR){//SI LE FIN DE COURSE ARRIERE N'EST PAS ACTIVER ALORS

```

```

    demarre_m_ar();//demare le moteur arrière
    flg_d_timer_1=0;//FLG_DEBUT_TIMER 1 MIS à 0
}
else if(E_CAME_EG_AR==0)//sinon si le fin de course arrière est activé alors on stop le
moteur
{
    stop_m_ar();//STOP Moteur arrière
    flg_d_timer_1=0;//FLG_DEBUT_TIMER 0 MIS à 0
}
}
if((ETAT==0 )) //MODE SELECTION AUCUN
{
    if(E_CAME_EG_AV){//SI LE FIN DE COURSE AVANT N'EST PAS ACTIVER ALORS
        demarre_m_av();//demare le moteur avant
        flg_d_timer_0=0;//FLG_DEBUT_TIMER 1 MIS à 0
    }
    else if(E_CAME_EG_AV==0)//sinon si le fin de course avant est activé alors on stop le moteur
    {
        stop_m_av();//STOP Moteur avant
        flg_d_timer_0=0;//FLG_DEBUT_TIMER 0 MIS à 0
    }
}
if(ETAT==1 )//MODE AUTOMATIQUE DONC que l'essuie glasse arrière
{
    if(E_CAME_EG_AV){//SI LE FIN DE COURSE AVANT N'EST PAS ACTIVER ALORS
        demarre_m_av();//demare le moteur avant
        flg_d_timer_0=0;//FLG_DEBUT_TIMER 0 MIS à 0
    }
    else if(E_CAME_EG_AV==0)//sinon si le fin de course avant est activé alors on stop le moteur
    {
        stop_m_av();//STOP Moteur
        flg_d_timer_0=0;//FLG_DEBUT_TIMER 0 MIS à 0
    }
}

if(E_COM_EG_AR==0)//SI LE MODE ESSUIE GLASSE ARRIERE EST ACTIVE ALORS FAIRE
{
    if(E_CAME_EG_AR){//SI LE FIN DE COURCE ARRIERE N'EST PAS ACTIVER ALORS
        demarre_m_ar();//demare le moteur arrière
        flg_d_timer_1=0;//FLG_DEBUT_TIMER 1 MIS à 0
    }
    else
    {
        if(fl_g_d_timer_1==0)//SI le flg_d_timer_1 est egale a zéro alors faire
        {
            flg_d_timer_1=1;//FLG_DEBUT_TIMER 1 MIS à 1
            set_timer1(2500);//On met le timer à 4ms
            overflow=0;
            stop_m_ar();//STOP le moteur ARRIERE
        }
    }
}
}
if(ETAT==2 || ETAT==4)//MODE LENT
{
    if(E_CAME_EG_AV){//SI LE FIN DE COURCE AVANT N'EST PAS ACTIVER ALORS
        demarre_m_av();//demare le moteur avant
        flg_d_timer_0=0;//FLG_DEBUT_TIMER 0 MIS à 0
    }
    else

```

```

    {
        if(flgs_d_timer_0==0)//SI le flgs_d_timer_0 est egale a zéro alors faire
        {
            flgs_d_timer_0=1;//FLG_DEBUT_TIMER 0 MIS à 1
            set_timer0(7812);//On met le timer à 0,4Secondes
            stop_m_av();//STOP le moteur AVANT
        }
    }
}

if(ETAT==5 )//Mode Automatique réglable par le potentiometre que a l'avant
{
    if(E_CAME_EG_AV){//SI LE FIN DE COURCE AVANT N'EST PAS ACTIVER ALORS
        demarre_m_av();//demare le moteur avant
        flgs_d_timer_0=0;//FLG_DEBUT_TIMER 0 MIS à 0
    }
    else
    {
        if(flgs_d_timer_0==0)//SI le flgs_d_timer_0 est egale a zéro alors faire
        {
            flgs_d_timer_0=1;//FLG_DEBUT_TIMER 0 MIS à 1
            set_timer0(READ_MCP3201()*16);//On met le timer en fonction du potentiometre
            stop_m_av();//STOP le moteur AVANT
        }
    }
}

if((ETAT==6 || ETAT==3))// Mode rapide ou Mode Automatique pluie forte
{
    if(E_CAME_EG_AV){//SI LE FIN DE COURCE AVANT N'EST PAS ACTIVER ALORS
        demarre_m_av();//demare le moteur avant
        flgs_d_timer_0=0;//FLG_DEBUT_TIMER 0 MIS à 0
    }
    else
    {
        if(flgs_d_timer_0==0)//SI le flgs_d_timer_0 est egale a zéro alors faire
        {
            flgs_d_timer_0=1;//FLG_DEBUT_TIMER 0 MIS à 1
            set_timer0(55770);//On met le timer a 2,9Seconde pour avoir un temps de 0,5S
            stop_m_av();//STOP le moteur AVANT
        }
    }
}

if((ETAT==2||ETAT==3||ETAT==4||ETAT==5||ETAT==6))//Pour l'essuie glasse arrière
{
    if(E_COM_EG_AR==0)//SI LE MODE ESSUIE GLASSE ARRIERE EST ACTIVE ALORS FAIRE
    {
        if(E_CAME_EG_AR){//SI LE FIN DE COURCE ARRIERE N'EST PAS ACTIVER ALORS
            demarre_m_ar();//demare le moteur arrière
            flgs_d_timer_1=0;//FLG_DEBUT_TIMER 1 MIS à 0
        }
        else
        {
            if(flgs_d_timer_1==0)//SI le flgs_d_timer_1 est egale a zéro alors faire
            {
                flgs_d_timer_1=1;//FLG_DEBUT_TIMER 1 MIS à 1
                set_timer1(2500);//On met le timer à 4ms
                overflow=0;
                stop_m_ar();//STOP le moteur ARRIERE
            }
        }
    }
}

```



```

    }
}

void main()
{
//SET TRIS
    Set_tris_a(0xFF);
    Set_tris_b(0b11011011);
    //set_tris_c(0b10111011); // car le LCD ne marche pas avec set_tris_c on a du mettre #use standard_io(C)
dans le .h
    set_tris_d(0x04);
    set_tris_e(0xFF);
//MOTEUR Eteint
    output_high(PWM_EG_AV);
    output_high(PWM_EG_AR);
    output_high(IN3);
    output_high(IN1);
//ADC
    output_high(MCP_CS); // arrêt de l'ADC externe
    output_low(MCP_CLK); // clk au NL0

    printf("début du programme \n\r");
//ADC
    setup_adc_ports(NO_ANALOGS, VSS_VDD); // Port A pas de analog
    setup_timer_0(RTCC_INTERNAL|RTCC_DIV_256); //3,4 s overflow
    setup_timer_1(T1_INTERNAL|T1_DIV_BY_8); //104 ms overflow
    setup_timer_3(T3_EXTERNAL | T3_DIV_BY_8); //104 ms overflow
    enable_interrupts(INT_TIMER0); // Active le timer 0
    enable_interrupts(INT_TIMER1); //Active le timer 1
    enable_interrupts(INT_TIMER3); //Active le timer 3
    ext_int_edge( 1, H_TO_L); // Front descendant sur IT externe 1
    enable_interrupts(INT_EXT1); // Active l'interruption Externe1

//CAN

    can_init();
    enable_interrupts(INT_CANIRX); // IT si message CAN invalide
    enable_interrupts(INT_CANERR); // IT si erreur bus CAN (plusieurs types d'erreurs possibles)
    enable_interrupts(INT_CANRX0); // IT sur réception d'un message dans le Buffer deréception n°0
//
    can_filtre(); // initialisation des valeurs des filtres et masques
    can_enable_filter(RXF0EN); // activation du filtre 0 pour le buffer de réception 0

    enable_interrupts(GLOBAL); // Active les IT globale

    GROVE_LCD_INIT();
    delay_ms(3000);
    GROVE_LCD_display();
    printf("Entrée dans le while true \n\r");
    GROVE_LCD_gotoxy(0,0);
    //printf(GROVE_LCD_printf,"DEBUT");

```

```
while(TRUE)
{
    Lecture();          //Image des entrée
    Automate();         //Gestion du mode état en fonction des entré ou des trames
    ecriture();         //Traitement de la variable état
}
```